PS3_LAU

Becky Lau

October 25, 2019

data munging

```
library(tidyverse)
```

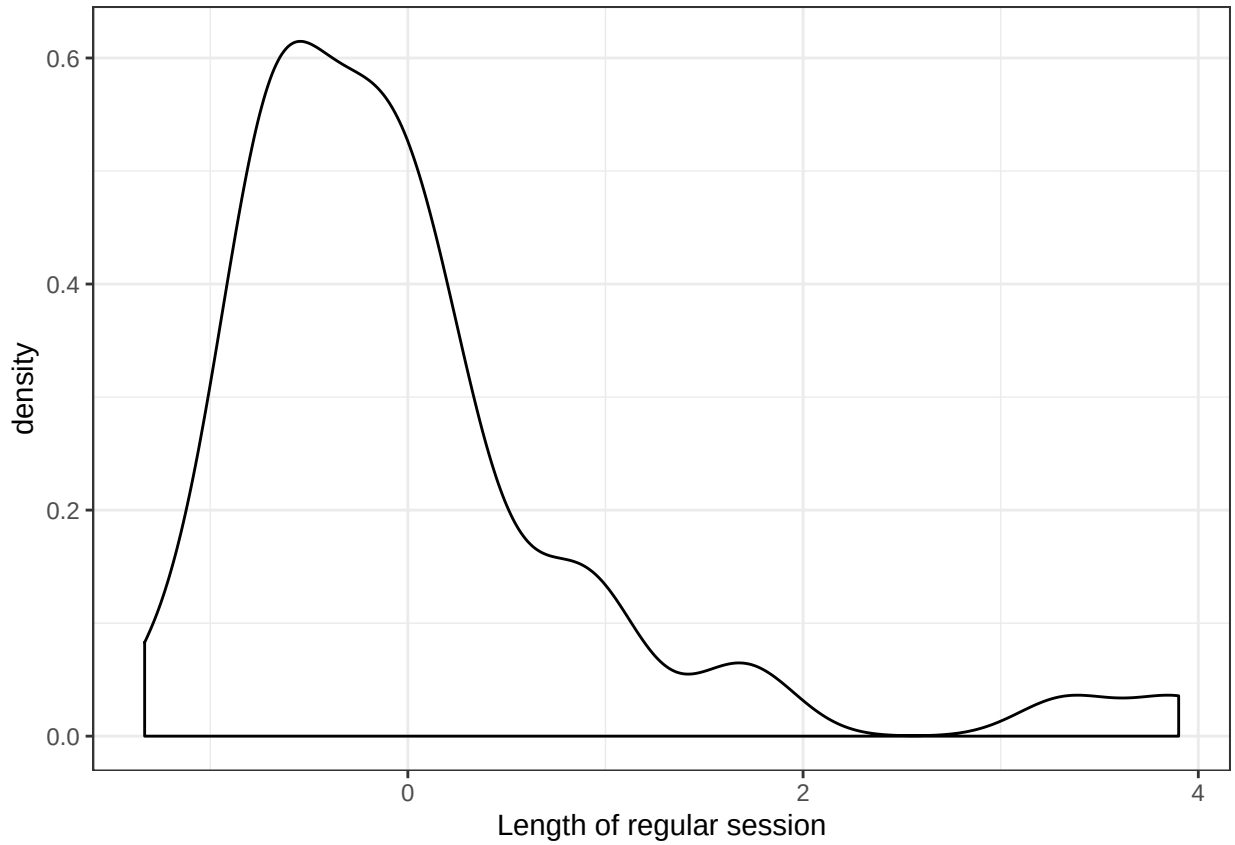
```
## -- Attaching packages ---------------------------------------------------- tidyverse 1.2.1 --

## v ggplot2 3.2.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## -- Conflicts ------------------------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
load("C:/Users/Keysar Lab/Box Sync/UML/legprof-components.v1.0.RData")
df <- x %>%
  select(state,sessid,slength,t_slength,salary_real,expend)%>%
  filter(sessid == "2009/10")%>%
  drop_na()%>%
  select(-sessid)%>%
  mutate_at(vars(slength,t_slength,salary_real,expend),~scale(.))
```

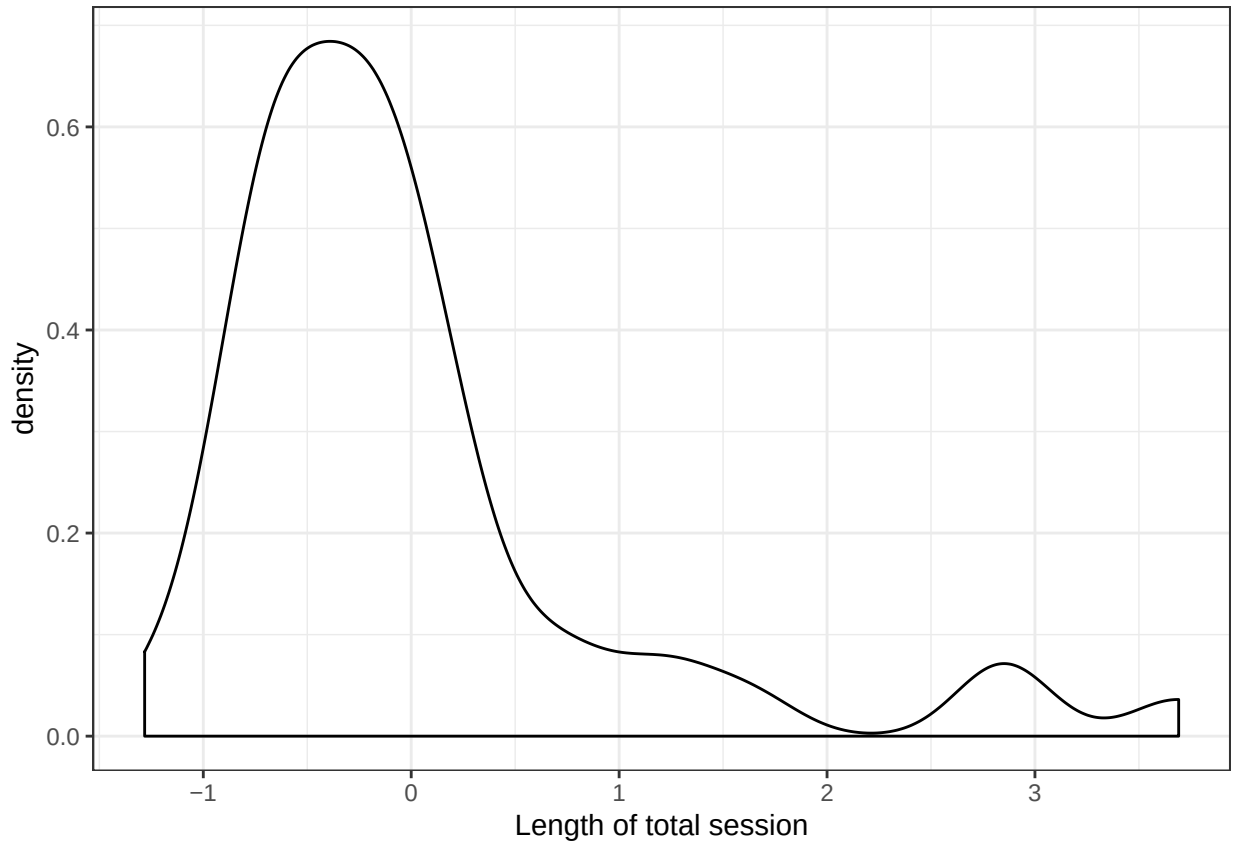
quick eda wtih density plots

All four variables show some level of right skewness. The tail is especially heavy for Expenditure. Some problematic outliers exist for Expenditure and Session Length (Total and Regular). All four variables seem unimodal in their distributions.

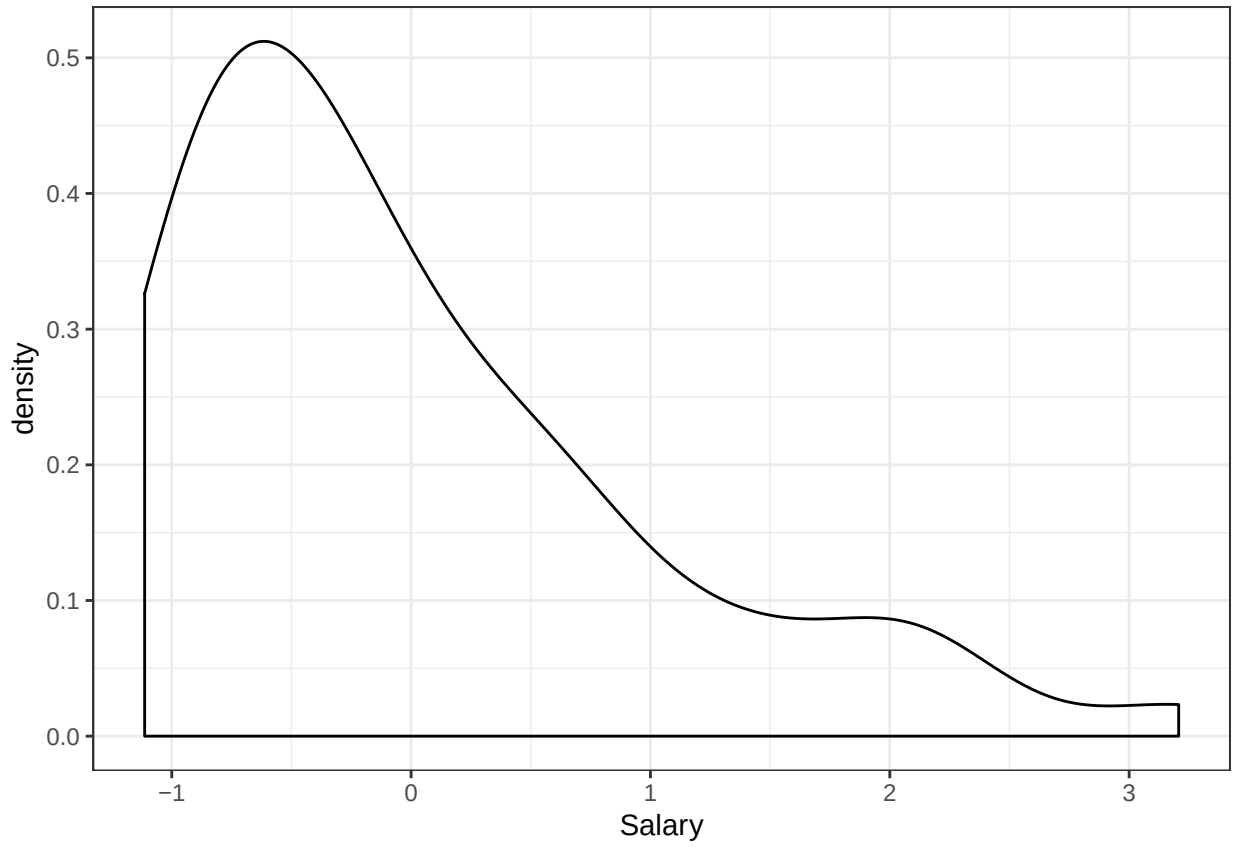
```
ggplot(df, aes(x = slength)) +
  geom_density() +
  theme_bw() +
  labs(x = "Length of regular session")
```



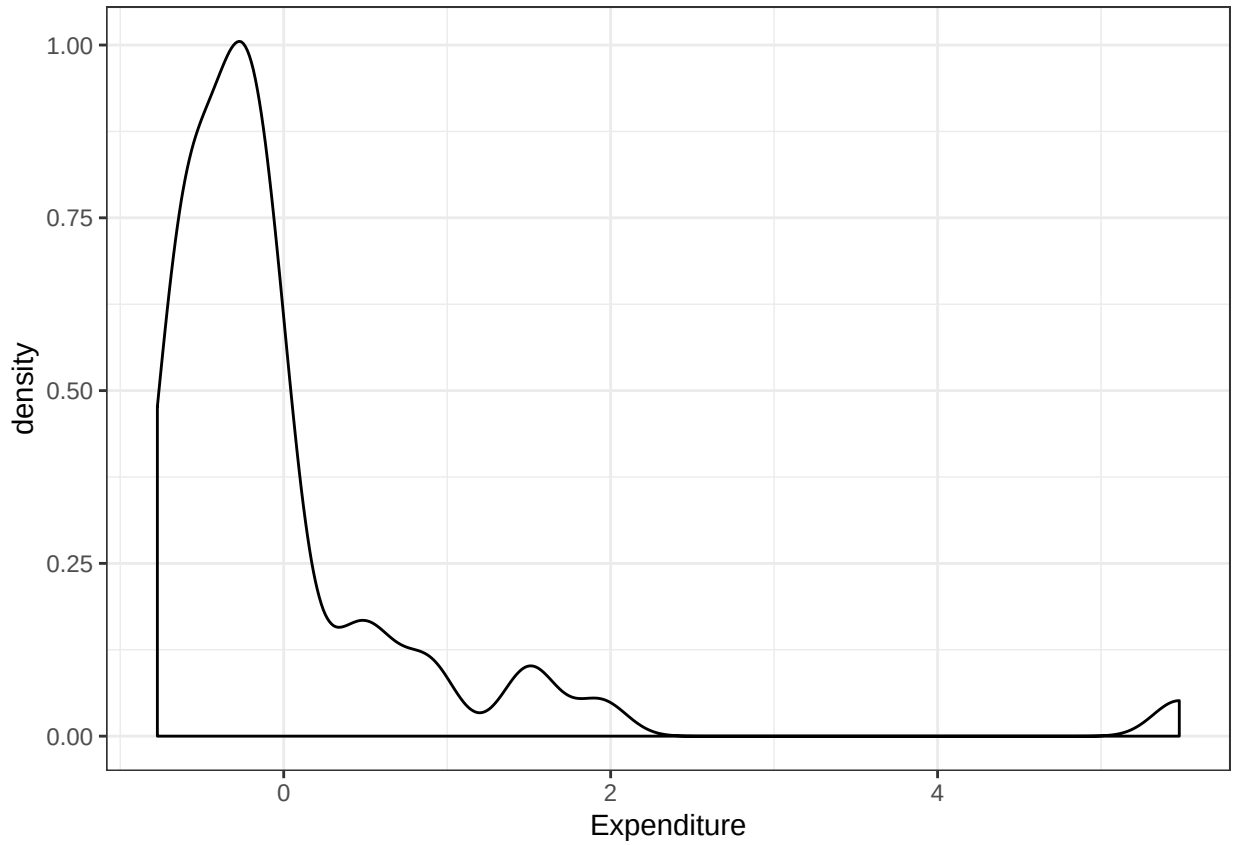

```
ggplot(df, aes(x = t_slength)) +
  geom_density() +
  theme_bw() +
  labs(x = "Length of total session")
```




```
ggplot(df, aes(x = salary_real)) +
  geom_density() +
  theme_bw() +
  labs(x = "Salary")
```

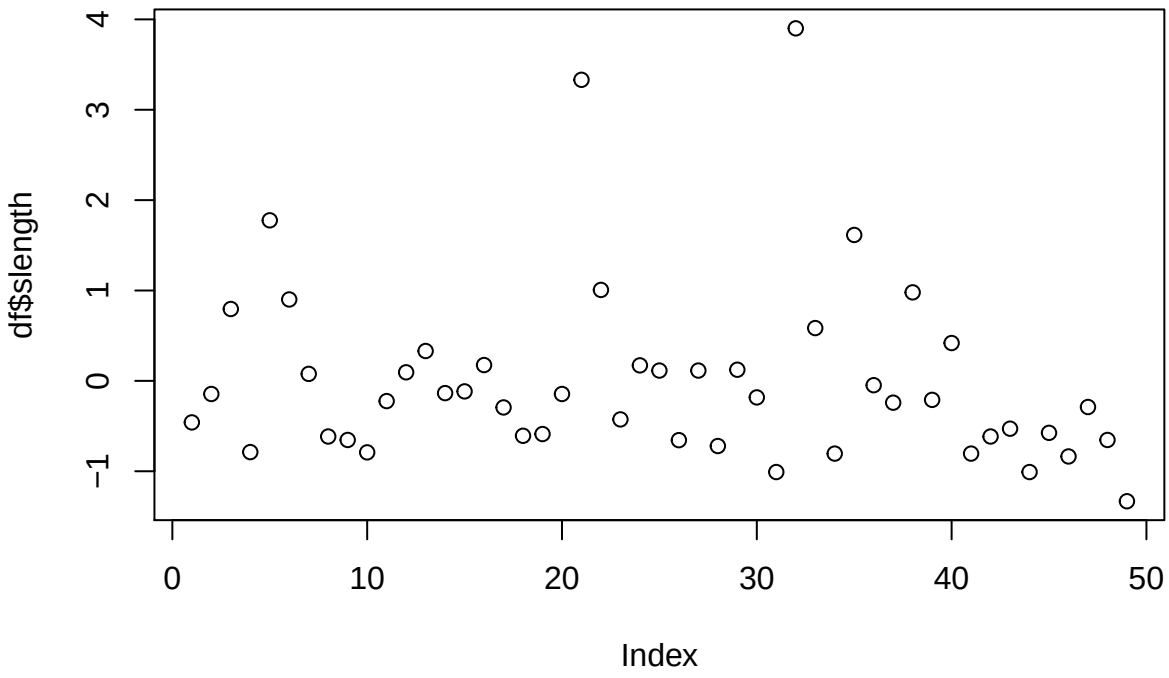



```
ggplot(df, aes(x = expend)) +
  geom_density() +
  theme_bw() +
  labs(x = "Expenditure")
```

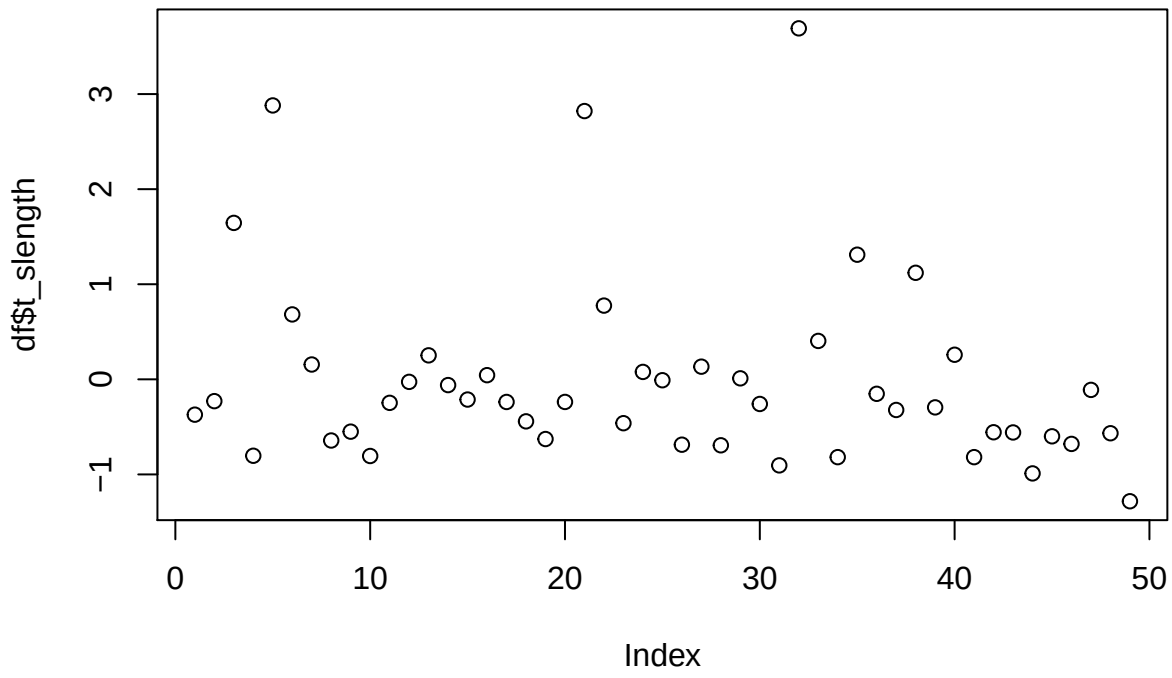


quick eda with scatterplots

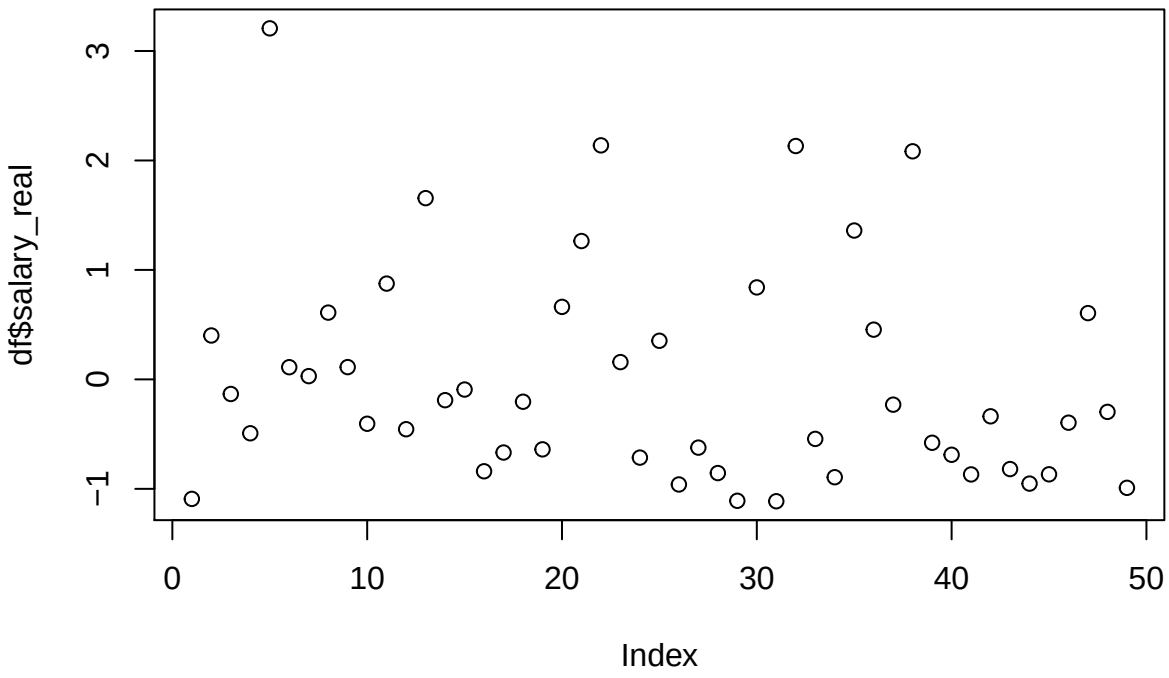
```
plot(df$slength)
```



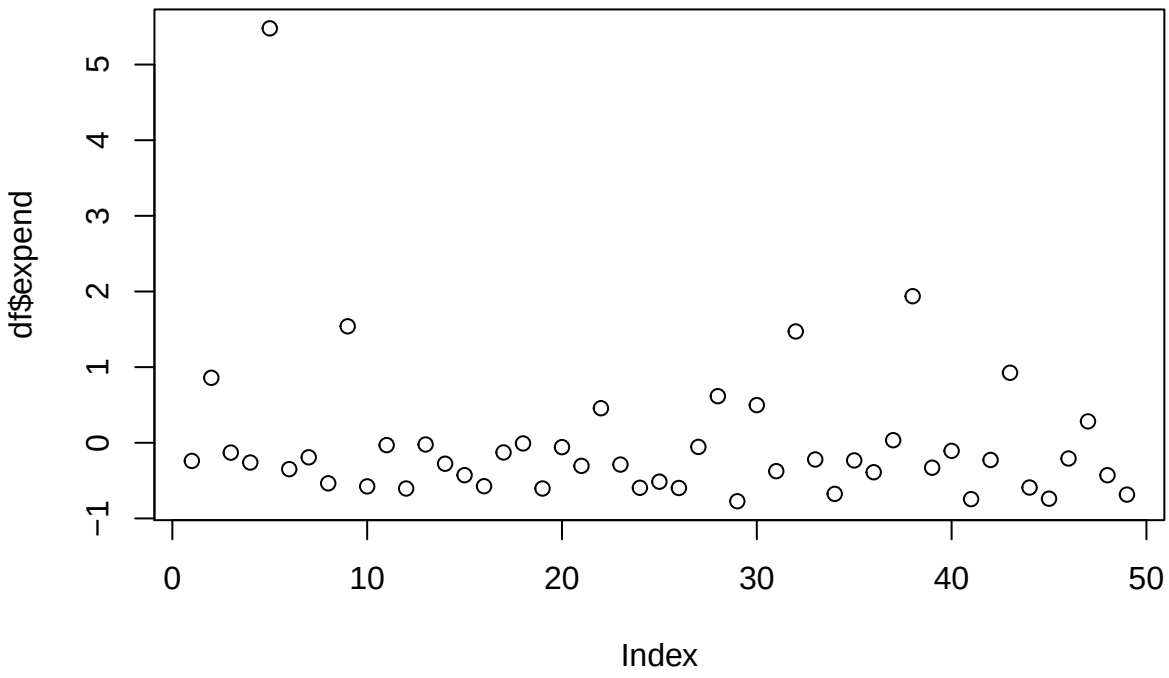

```
plot(df$t_slength)
```




```
plot(df$salary_real)
```




```
plot(df$expend)
```



diagnosing clusterability

It seems that the s_length and salary may be most clusterable according to a visual analysis of the ODI plot and scatterplots. However, generally the likelihood that natural non-random structure in the data existing is low.

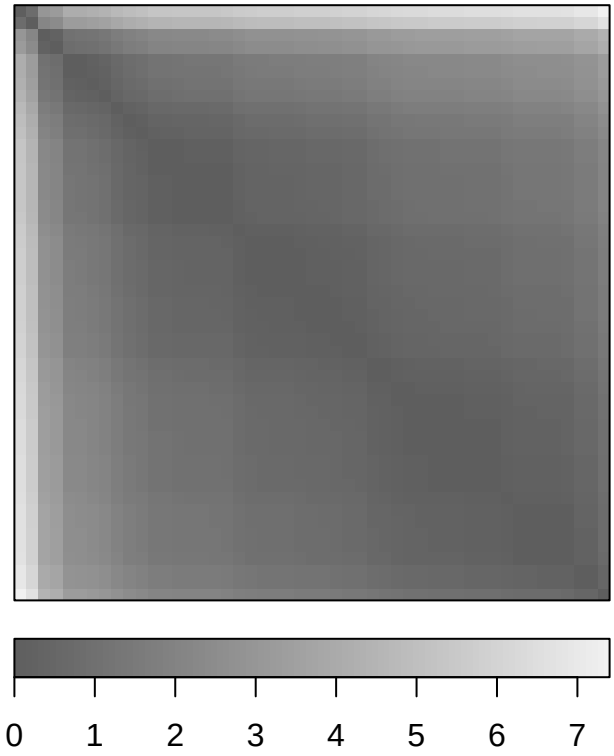
Because s_length seems most clusterable, analysis below focuses on session length

```
s_length <- df %>% select(state, slength)
slength_dist <- dist(s_length)
```

```
## Warning in dist(s_length): NAs introduced by coercion
```

```
seriation::dissplot(slength_dist)
```

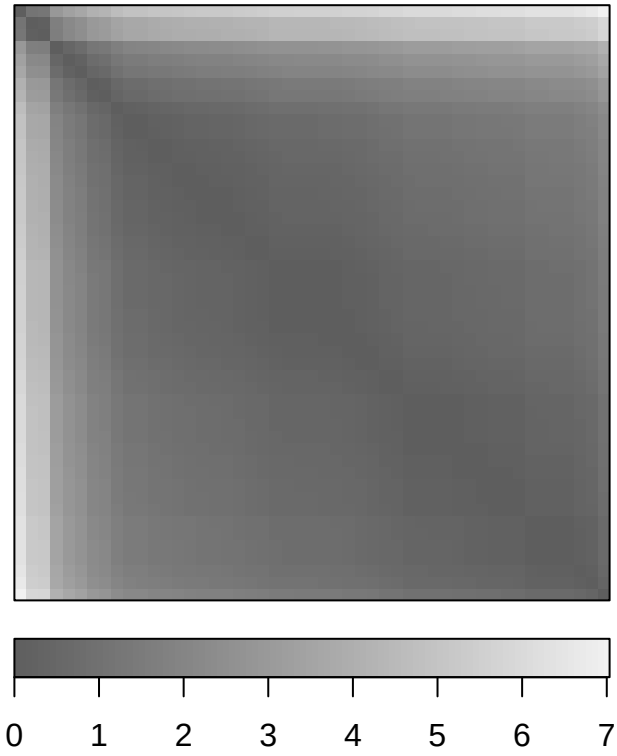
```
## Registered S3 method overwritten by 'seriation':
##   method         from
##   reorder.hclust gclus
```




```
ts_length <- df %>% select(state, t_slength)
tslength_dist <- dist(ts_length)
```

```
## Warning in dist(ts_length): NAs introduced by coercion
```

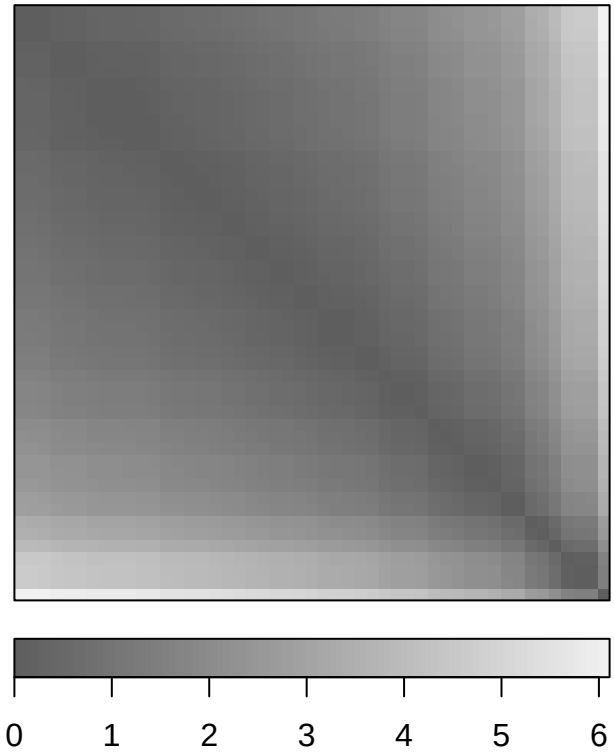
```
seriation::dissplot(tslength_dist)
```




```
salary <- df %>% select(state, salary_real)
salary_dist <- dist(salary)
```

```
## Warning in dist(salary): NAs introduced by coercion
```

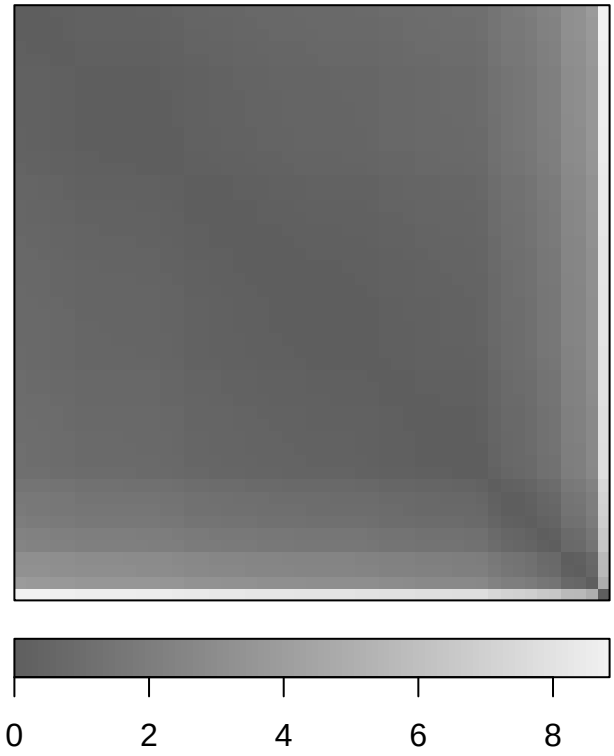
```
seriation::dissplot(salary_dist)
```




```
expend <- df %>% select(state, expend)
expend_dist <- dist(expend)
```

```
## Warning in dist(expend): NAs introduced by coercion
```

```
seriation::dissplot(expend_dist)
```



k means - using slength

Only four states (CA, MA, NY, OH) are classified into the first cluster. However, this seems to be driven by NY and MA being outliers.

```
# fit the algorithm
set.seed(634)

kmeans <- kmeans(df$slength,
                 centers = 2,
                 nstart = 15)

# Inspect the kmeans object
str(kmeans)
```

```
## List of 9
##  $ cluster     : int [1:49] 2 2 1 2 1 1 2 2 2 2 ...
##  $ centers     : num [1:2, 1] 1.788 -0.349
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "1" "2"
##   .. ..$ : NULL
##  $ totss       : num 48
##  $ withinss    : num [1:2] 9.91 7.52
##  $ tot.withinss: num 17.4
##  $ betweenss   : num 30.6
```

```
##  $ size        : int [1:2] 8 41
##  $ iter        : int 1
##  $ ifault      : int 0
##  - attr(*, "class")= chr "kmeans"
```

```
# Or call individual values, such as...
kmeans$cluster
```

```
##  [1] 2 2 1 2 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 2 2 2 2 2 2 2 2 2 1 2 2 1
## [36] 2 2 1 2 2 2 2 2 2 2 2 2 2 2
```

```
kmeans$centers
```

```
##         [,1]
## 1  1.7881137
## 2 -0.3489002
```

```
kmeans$size
```

```
## [1]  8 41
```

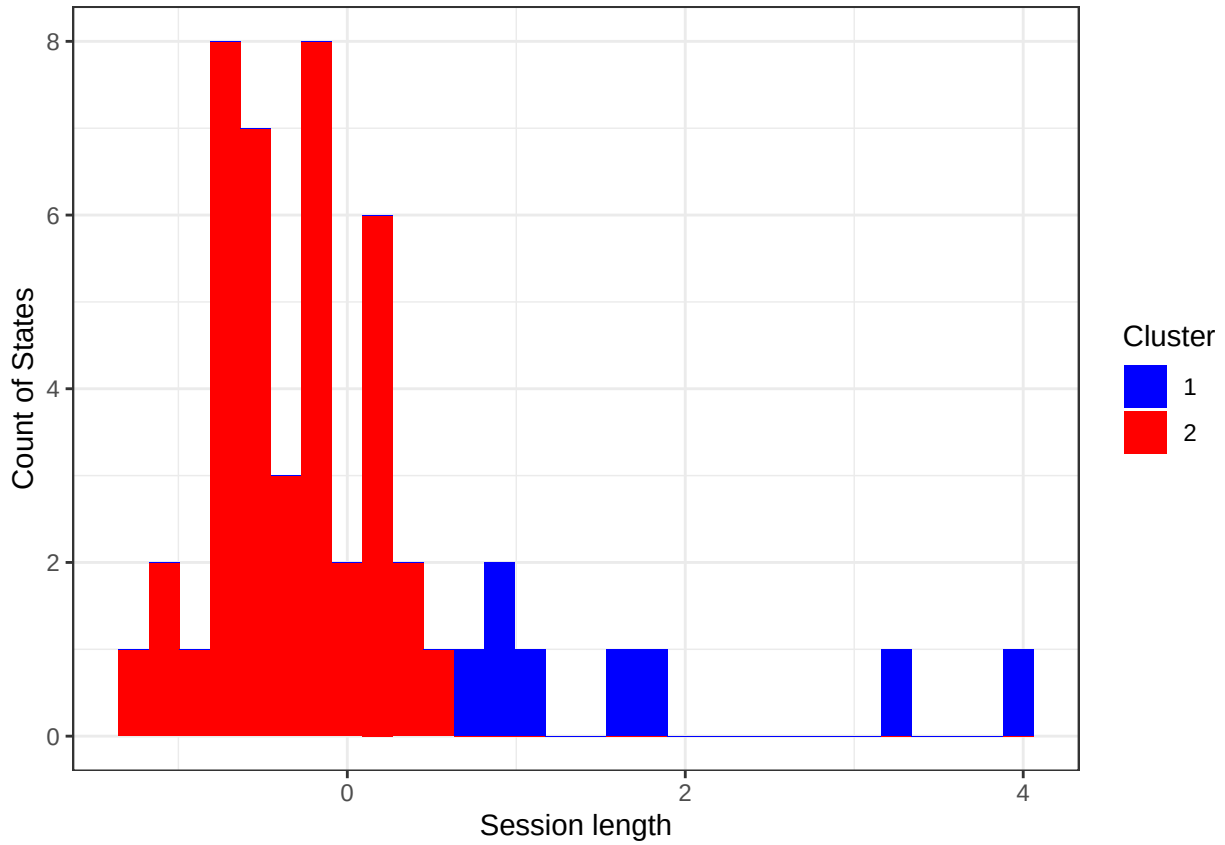
```
df$Cluster <- as.factor(kmeans$cluster) # save clusters in df

# Assess a little more descriptively
t <- as.table(kmeans$cluster)
t <- data.frame(t)
rownames(t) <- df$state
colnames(t)[colnames(t)=="Freq"] <- "Assignment"
t$Var1 <- NULL

print(t)
```

```
##                Assignment
## Alabama                 2
## Alaska                  2
## Arizona                 1
## Arkansas                2
## California              1
## Colorado                1
## Connecticut             2
## Delaware                2
## Florida                 2
## Georgia                 2
## Hawaii                  2
## Idaho                   2
## Illinois                2
## Indiana                 2
## Iowa                    2
## Kansas                  2
## Kentucky                2
## Louisiana               2
```

```
## Maine                2
## Maryland             2
## Massachusetts        1
## Michigan             1
## Minnesota            2
## Mississippi          2
## Missouri             2
## Montana              2
## Nebraska             2
## Nevada               2
## New Hampshire        2
## New Jersey           2
## New Mexico           2
## New York             1
## North Carolina       2
## North Dakota         2
## Ohio                 1
## Oklahoma             2
## Oregon               2
## Pennsylvania         1
## Rhode Island         2
## South Carolina       2
## South Dakota         2
## Tennessee            2
## Texas                2
## Utah                 2
## Vermont              2
## Virginia             2
## Washington           2
## West Virginia        2
## Wyoming              2
```

```
ggplot(df, aes(slength, fill = Cluster)) +
  geom_histogram() +
  theme_bw() +
  scale_fill_manual(values=c("blue", "red")) +
  labs(x = "Session length",
       y = "Count of States")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



k means - using slength and remove NY and MA

```
# fit the algorithm

df_no_NY_MA <- df %>% filter(state!="New York",state!="Massachusetts")

set.seed(634)

kmeans <- kmeans(df_no_NY_MA$slength,
                 centers = 2,
                 nstart = 15)

# Inspect the kmeans object
str(kmeans)
```

```
## List of 9
##  $ cluster     : int [1:47] 2 2 1 2 1 1 2 2 2 2 ...
##  $ centers     : num [1:2, 1] 0.934 -0.412
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "1" "2"
##   .. ..$ : NULL
##  $ totss       : num 20.6
##  $ withinss    : num [1:2] 1.95 5.45
##  $ tot.withinss: num 7.4
```

```
##  $ betweenss    : num 13.2
##  $ size         : int [1:2] 9 38
##  $ iter         : int 1
##  $ ifault       : int 0
##  - attr(*, "class")= chr "kmeans"
```

```
# Or call individual values, such as...
kmeans$cluster
```

```
##  [1] 2 2 1 2 1 1 2 2 2 2 2 2 1 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 1 2 1 2 2
## [36] 1 2 1 2 2 2 2 2 2 2 2 2
```

```
kmeans$centers
```

```
##         [,1]
## 1  0.9340324
## 2 -0.4115341
```

```
kmeans$size
```

```
## [1]  9 38
```

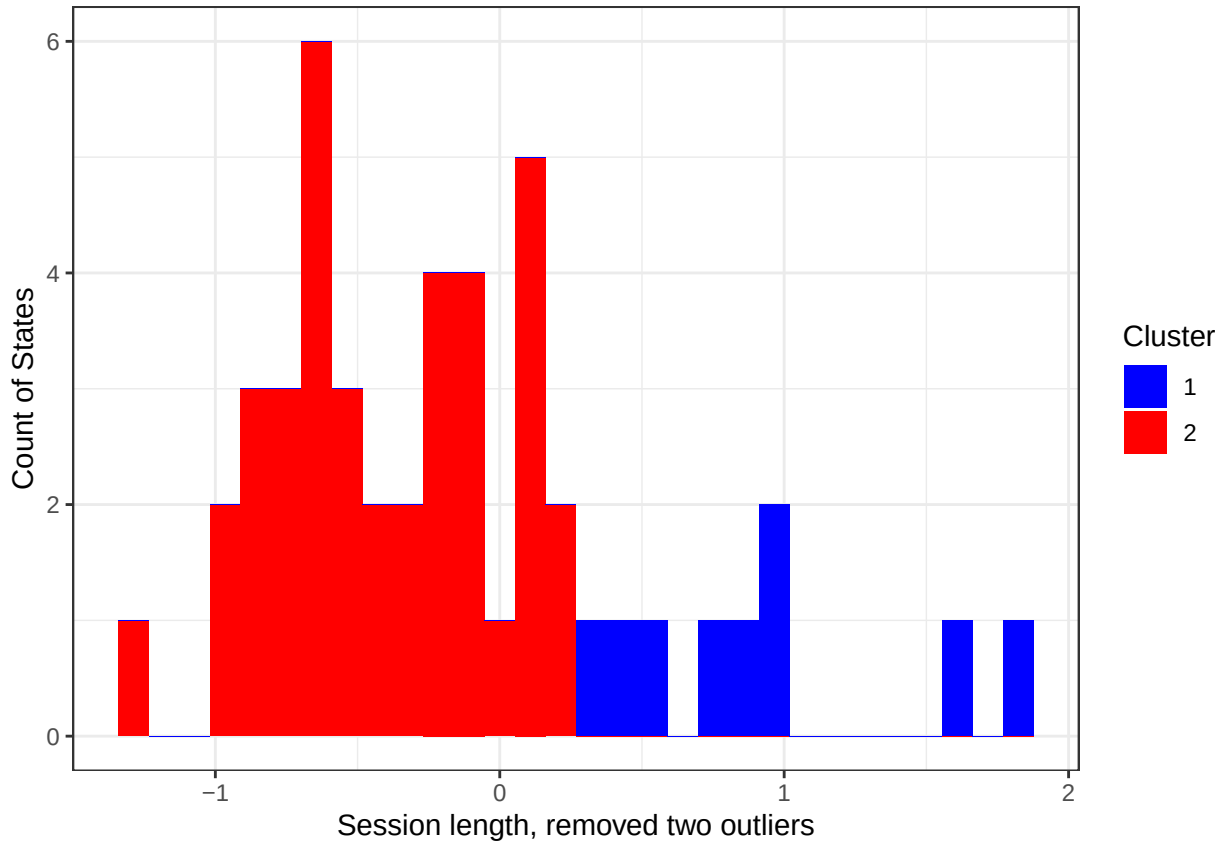
```
df_no_NY_MA$Cluster <- as.factor(kmeans$cluster) # save clusters in df

# Assess a little more descriptively
t <- as.table(kmeans$cluster)
t <- data.frame(t)
rownames(t) <- df_no_NY_MA$state
colnames(t)[colnames(t)=="Freq"] <- "Assignment"
t$Var1 <- NULL

print(t)
```

```
##                Assignment
## Alabama                 2
## Alaska                  2
## Arizona                 1
## Arkansas                2
## California              1
## Colorado                1
## Connecticut             2
## Delaware                2
## Florida                 2
## Georgia                 2
## Hawaii                  2
## Idaho                   2
## Illinois                1
## Indiana                 2
## Iowa                    2
## Kansas                  2
## Kentucky                2
```

```
## Louisiana              2
## Maine                  2
## Maryland               2
## Michigan               1
## Minnesota              2
## Mississippi            2
## Missouri               2
## Montana                2
## Nebraska               2
## Nevada                 2
## New Hampshire          2
## New Jersey             2
## New Mexico             2
## North Carolina         1
## North Dakota           2
## Ohio                   1
## Oklahoma               2
## Oregon                 2
## Pennsylvania           1
## Rhode Island           2
## South Carolina         1
## South Dakota           2
## Tennessee              2
## Texas                  2
## Utah                   2
## Vermont                2
## Virginia               2
## Washington             2
## West Virginia          2
## Wyoming                2
```

```
ggplot(df_no_NY_MA, aes(slength, fill = Cluster)) +
  geom_histogram() +
  theme_bw() +
  scale_fill_manual(values=c("blue", "red")) +
  labs(x = "Session length, removed two outliers",
       y = "Count of States")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



gaussian EM - using slength

One of the clusters is driven mostly by outliers.

```
library(mixtools) # fitting GMMs via EM
```

```
## mixtools package, version 1.1.0, Released 2017-03-10
## This package is based upon work supported by the National Science Foundation under Grant No. SES-0518
```

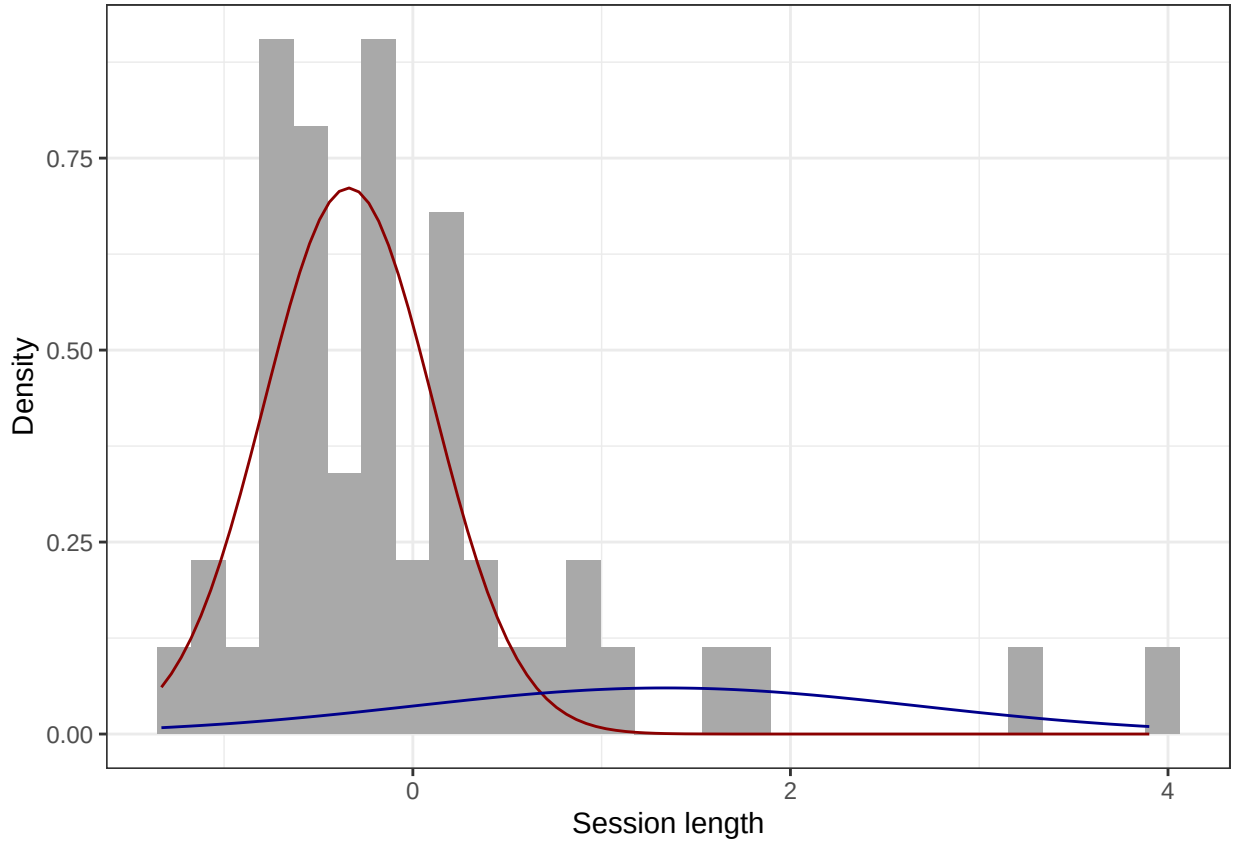
```
library(plotGMM) # customizing GMM plot

set.seed(7355)
gmm1 <- normalmixEM(df$slength,
            k = 2) # fit the GMM using EM and 2 comps
```

```
## number of iterations= 54
```

```
ggplot(data.frame(x = gmm1$x)) +
  geom_histogram(aes(x, ..density..), fill = "darkgray") +
  stat_function(geom = "line", fun = plot_mix_comps,
                args = list(gmm1$mu[1], gmm1$sigma[1], lam = gmm1$lambda[1]),
                colour = "darkred") +
  stat_function(geom = "line", fun = plot_mix_comps,
```

```
                args = list(gmm1$mu[2], gmm1$sigma[2], lam = gmm1$lambda[2]),
                colour = "darkblue") +
  xlab("Session length") +
  ylab("Density") +
  theme_bw()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



gaussian EM - using slength

Removing the outliers and refitting. Now the clusters look a bit more distinct.

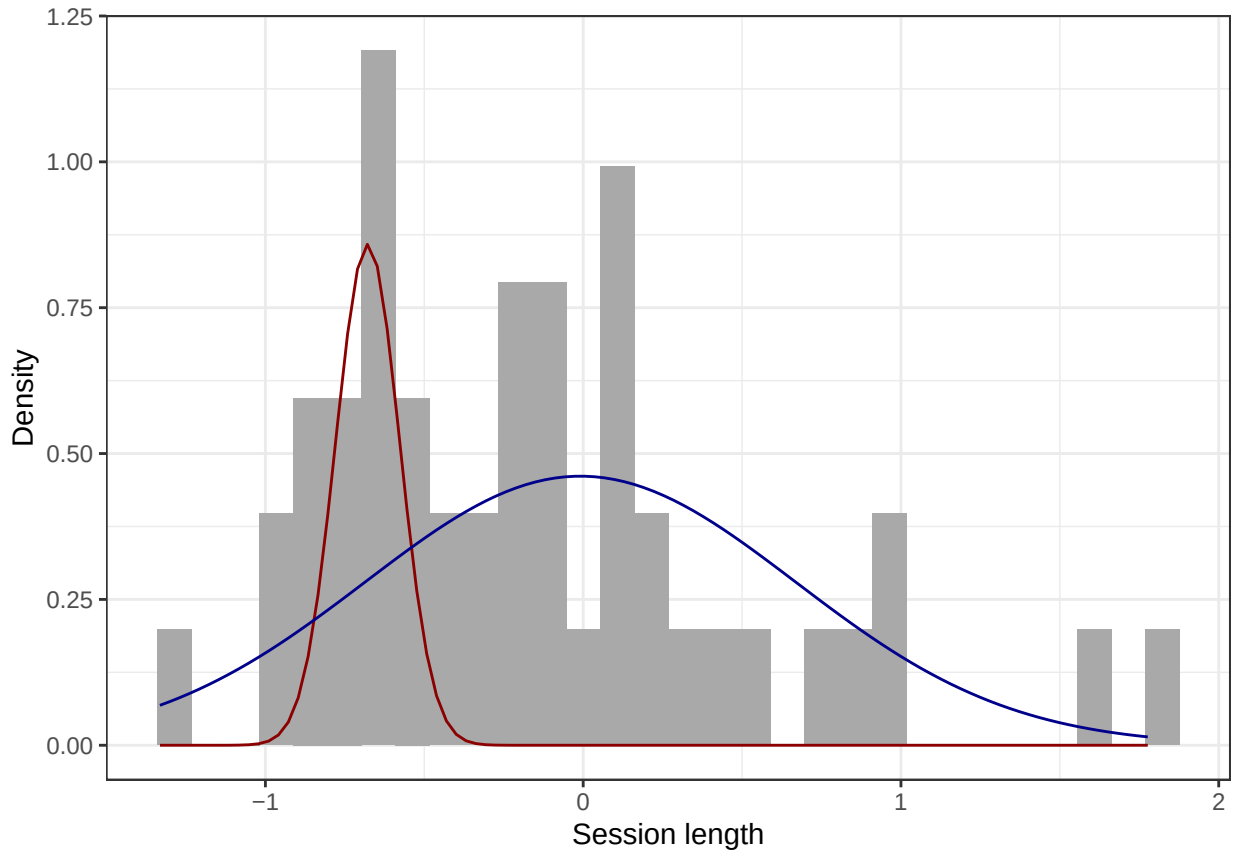
```
set.seed(7355)
gmm1 <- normalmixEM(df_no_NY_MA$slength,
            k = 2) # fit the GMM using EM and 2 comps
```

```
## number of iterations= 33
```

```
ggplot(data.frame(x = gmm1$x)) +
  geom_histogram(aes(x, ..density..), fill = "darkgray") +
  stat_function(geom = "line", fun = plot_mix_comps,
                args = list(gmm1$mu[1], gmm1$sigma[1], lam = gmm1$lambda[1]),
                colour = "darkred") +
```

```
  stat_function(geom = "line", fun = plot_mix_comps,
                args = list(gmm1$mu[2], gmm1$sigma[2], lam = gmm1$lambda[2]),
                colour = "darkblue") +
  xlab("Session length") +
  ylab("Density") +
  theme_bw()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



trying out fcm on slength data without outliers

The output allows us to assess the degree to which each observation is assigned to each cluster. In FCM the distribution of states into the clusters seem a bit more even.

```
e1071::cmeans(df_no_NY_MA$slength, centers = 2,verbose = TRUE,
       dist = "euclidean", method = "cmeans")
```

```
## Iteration:   1, Error:  0.1467938351
## Iteration:   2, Error:  0.1286189850
## Iteration:   3, Error:  0.1273816431
## Iteration:   4, Error:  0.1272138359
## Iteration:   5, Error:  0.1271358319
## Iteration:   6, Error:  0.1270905029
```

```
## Iteration:   7, Error:  0.1270633871
## Iteration:   8, Error:  0.1270469621
## Iteration:   9, Error:  0.1270369254
## Iteration:  10, Error:  0.1270307520
## Iteration:  11, Error:  0.1270269359
## Iteration:  12, Error:  0.1270245680
## Iteration:  13, Error:  0.1270230943
## Iteration:  14, Error:  0.1270221750
## Iteration:  15, Error:  0.1270216006
## Iteration:  16, Error:  0.1270212411
## Iteration:  17, Error:  0.1270210159
## Iteration:  18, Error:  0.1270208747
## Iteration:  19, Error:  0.1270207861
## Iteration:  20, Error:  0.1270207304
## Iteration:  21, Error:  0.1270206955
## Iteration:  22, Error:  0.1270206736
## Iteration:  23, Error:  0.1270206598
## Iteration:  24, Error:  0.1270206511
## Iteration:  25, Error:  0.1270206456
## Iteration:  26, Error:  0.1270206422
## Iteration:  27, Error:  0.1270206401
## Iteration:  28 converged, Error:  0.1270206387

## Fuzzy c-means clustering with 2 clusters
##
## Cluster centers:
##         [,1]
## 1  0.7455732
## 2 -0.5033660
##
## Memberships:
##                   1           2
##  [1,] 0.0013250200 0.998674980
##  [2,] 0.1391427295 0.860857270
##  [3,] 0.9985418778 0.001458122
##  [4,] 0.0333332301 0.966666770
##  [5,] 0.8302068579 0.169793142
##  [6,] 0.9879147105 0.012085290
##  [7,] 0.4301010582 0.569898942
##  [8,] 0.0067504619 0.993249538
##  [9,] 0.0114283121 0.988571688
## [10,] 0.0339070409 0.966092959
## [11,] 0.0769032903 0.923096710
## [12,] 0.4577519968 0.542248003
## [13,] 0.8014540206 0.198545979
## [14,] 0.1483144989 0.851685501
## [15,] 0.1676893771 0.832310623
## [16,] 0.5853908675 0.414609132
## [17,] 0.0390205380 0.960979462
## [18,] 0.0058850089 0.994114991
## [19,] 0.0040600951 0.995939905
## [20,] 0.1391427295 0.860857270
## [21,] 0.9710348097 0.028965190
## [22,] 0.0042992028 0.995700797
```

```
## [23,] 0.5800913210 0.419908679
## [24,] 0.4890608347 0.510939165
## [25,] 0.0117377477 0.988262252
## [26,] 0.4890608347 0.510939165
## [27,] 0.0215473308 0.978452669
## [28,] 0.5035606390 0.496439361
## [29,] 0.1063099098 0.893690090
## [30,] 0.0766575996 0.923342400
## [31,] 0.9784214165 0.021578583
## [32,] 0.0364248158 0.963575184
## [33,] 0.8559168472 0.144083153
## [34,] 0.2478115638 0.752188436
## [35,] 0.0659362870 0.934063713
## [36,] 0.9757557278 0.024244272
## [37,] 0.0860591865 0.913940813
## [38,] 0.8884262015 0.111573799
## [39,] 0.0364248158 0.963575184
## [40,] 0.0067504619 0.993249538
## [41,] 0.0004067492 0.999593251
## [42,] 0.0766575893 0.923342411
## [43,] 0.0029363212 0.997063679
## [44,] 0.0423902318 0.957609768
## [45,] 0.0410882732 0.958911727
## [46,] 0.0114283121 0.988571688
## [47,] 0.1372313336 0.862768666
##
## Closest hard clustering:
##  [1] 2 2 1 2 1 1 2 2 2 2 2 2 1 2 2 1 2 2 2 2 1 2 1 2 2 2 2 1 2 2 1 2 1 2 2
## [36] 1 2 1 2 2 2 2 2 2 2 2 2
##
## Available components:
## [1] "centers"     "size"        "cluster"     "membership"  "iter"
## [6] "withinerror" "call"
```

validation

I'm not sure why I keep getting this error about row names not specified.

```
df_int <- as.matrix(df_no_NY_MA$slength)
internal <- clValid::clValid(df_int, nClust = 2:10, #values of k from 2 to 10, do validation
                    clMethods = "kmeans",
                    validation = "internal")
```

```
## Warning in clValid::clValid(df_int, nClust = 2:10, clMethods = "kmeans", :
## rownames for data not specified, using 1:nrow(data)
```

```
summary(internal)
```

```
##  Length   Class    Mode
##       1 clValid      S4
```